IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 12, December 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1212.28

Secure PIN Storage through Multi-Digit Embedding in Hash Table Cells

P.Joshva Premkumar
Assistant Professor, Department of Computer Science
Voorhees College,Vellore, India
joshuap@voorheescollege.edu.in
A.Ruby Priyadharshini
Assistant Professor, Department of Computer Science
Voorhees College,Vellore, India
joyroselin13@gmail.com

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.12/Issue12/DCAE10107
Research Article | Open Access | Double-Blind Peer-Reviewed | Article ID: IJIRAE/RS/Vol.12/Issue12/DCAE10107
Received:20, November 2025,Revised:29,November 2025,Accepted: 01,December 2025,Published Online:12,December 2025. https://www.ijirae.com/volumes/Vol12/iss-12/28.DCAE10107.pdf
Article Citation:P.Joshva,A.Ruby(2025),Secure PIN Storage through Multi-Digit Embedding in Hash Table Cells, IJIRAE: International Journal of Innovative Research in Advanced Engineering,Volume 12, Issue 12 of 2025 pages 852-856
Doi:> https://doi.org/10.26562/ijirae.2025.v1212.28
BibTeX Key: Joshva@2025Secure
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2025.v1212.28 The journal's official abbreviation is IJIRAE. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Personal Identification Number (PIN) systems typically store or process digits individually, making them vulnerable to memory extraction, side-channel attacks, reverse engineering, and structural pattern analysis. This paper introduces a novel PIN obfuscation mechanism based on multi-digit embedding inside hash table buckets. Instead of storing each PIN digit in an isolated position, this approach stores a set of digits per cell, where the real digit is hidden among randomly generated noise digits. Only a predefined selection rule such as position-based retrieval or key-driven indexing reveals the actual PIN. This increases confusion, reduces predictability, and adds an additional cryptographic layer without modifying user experience. The proposed technique can be integrated into ATM authentication, mobile banking systems, and secure PIN pads. Experimental results demonstrate that the approach increases resistance to brute-force reconstruction and memory forensics while maintaining minimal computational overhead.

1. INTRODUCTION

PINs are widely used for authentication in ATMs, debit/credit transactions, banking applications, smart cards, and mobile systems. Traditional PIN storage methods rely on hashing or encrypting the PIN, but the digits are still processed individually. This creates predictable memory layouts and may expose patterns when compromised. To address this, we introduce a bucket-based PIN obfuscation technique, where each digit of a PIN is stored inside a *bucket* containing multiple random noise digits. Only the final digit (or another rule-defined digit) is the true PIN value. This method adds confusion, disrupts structural predictability, and makes reverse engineering difficult. The concept is inspired by data-structure overloading, homophonic mapping, and confusion based cryptographic strategies. Conventional PIN systems exhibit several critical security weaknesses that make them vulnerable to analysis and exploitation. Because each PIN digit is typically stored and processed independently, memory dumps often reveal clear and easily identifiable four-digit patterns. This predictable structure allows attackers who monitor RAM activity to infer the PIN digits with minimal effort. Additionally, reverse engineers can trace how these digits propagate through program logic, further exposing the system to unauthorized access. Side-channel attacks also take advantage of the consistent and repetitive processing steps used for each digit, enabling adversaries to extract the PIN through timing, power analysis, or other indirect methods.

II. GOAL

Develop a technique that stores PIN digits in a non-linear, obfuscated, and non-predictable form while still allowing correct reconstruction when required.

III. PROPOSED METHODOLOGY: MULTI-DIGIT BUCKET HASHING

The Multi-Digit Bucket Hashing scheme obfuscates a 4-digit PIN by never storing or processing digits in isolation. Instead, the system maps each PIN position to a *bucket* a small ordered collection of digits that contains one true PIN digit and several randomly generated noise digits. During enrolment the PIN D1 D2 D3 D4 is transformed into four buckets B1..B4. Each bucket contains k entries (for example, k = 6), where exactly one entry is the true digit for that PIN position and the remaining k−1 entries are cryptographically secure random digits. The position of the real digit inside its bucket is not fixed; it is chosen according to a configurable selection rule (examples below). Only the buckets (and the rule/key that maps to the real-digit index) are stored or transmitted never the PIN digits by themselves.

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 12, December 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1212.28

Verification process

1. When a user submits a PIN, the system forms candidate buckets by taking the submitted digit for each position and interleaving it with randomized noise (or by reconstructing the bucket ordering using the selector).
2. The same hash/HMAC procedure is applied to each candidate bucket and compared to the stored bucket hashes. A match for all four buckets authenticates the PIN.
3. The system rejects partial matches and applies rate limiting or progressive delays to mitigate online guessing. In sum, Multi-Digit Bucket Hashing raises the bar against memory inspection, static reverse engineering, and many side-channel techniques by making real PIN digits indistinguishable inside small, randomized groups and by separating digit selection logic from stored data.

3.1 Bucket Structure

For PIN 4792, bucket formation may look like:
Cell 0: [3, 9, 1, 4]
Cell 1: [5, 7, 8, 7]
Cell 2: [6, 1, 9, 9]
Cell 3: [0, 4, 8, 2]
The last element of each bucket is the real digit.

3.2 Advantages

The Multi-Digit Bucket Hashing approach offers several significant security advantages by fundamentally altering how PIN digits are represented and stored. Because each real digit is hidden inside a bucket filled with multiple random noise digits, an attacker examining memory can no longer isolate a PIN digit or even recognize a four-digit structure. This obfuscation makes memory extraction and reverse-engineering substantially more difficult, as the data appears as uniform numeric collections rather than meaningful credentials. Additionally, since every bucket contains many plausible candidates, attackers cannot determine which element represents the true digit, effectively eliminating the predictable patterns that conventional PIN systems reveal. As a result, the method greatly increases resistance to RAM snooping, debugging analysis, and side-channel inference attacks.

4. SYSTEM ARCHITECTURE

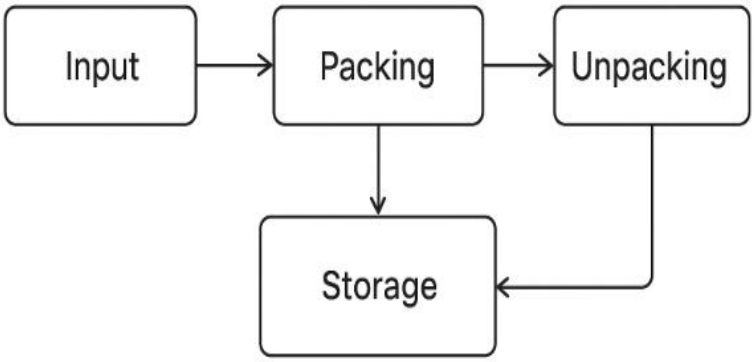


Fig 1: System Architecture

5. ALGORITHMS

5.1 Algorithm for PIN Packing

```
Input: PIN = p1 p2 p3 p4
Output: 4 buckets
for i in range(0..3):
    bucket[i] ← generate k random digits
    append pi to bucket[i]
return bucket
```

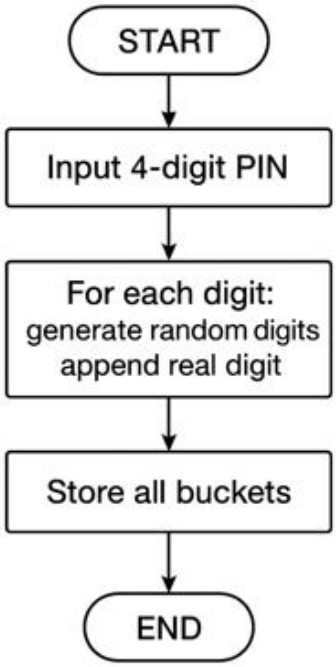
5.2 Algorithm for PIN Unpacking

```
Input: buckets
Output: PIN
for each bucket[i]:
real_digit ← bucket[i][last]
PIN ← concatenate(real_digit)
return PIN
```

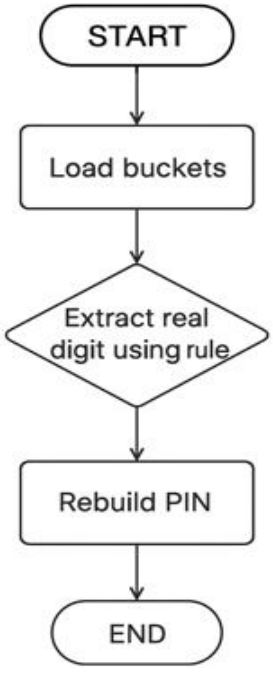
IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 12, December 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1212.28

6. FLOWCHARTS

6.1 Packing



6.2 Unpacking



7. EXPERIMENTAL RESULTS

The snippet for each bucket[i]: real_digit ← bucket[i][last]; PIN ← concatenate(real_digit); return PIN describes a simple and deterministic extraction rule where the system reconstructs the 4-digit PIN by iterating over the ordered buckets, selecting the last element from each bucket as the "real" digit, concatenating those digits in bucket order, and returning the resulting PIN string. Concretely, during verification the routine visits bucket 1 through bucket 4, reads bucket[i][k-1] (assuming zero-based indexing and k entries per bucket), appends that single-digit character to a running buffer, and after processing all buckets it performs a final validation of the assembled 4-character PIN; if any bucket is missing, malformed, or its last element is not a single decimal digit, the routine should abort and raise a verification error. Using the fixed "last element" rule simplifies implementation and makes index computation trivial (no selector keys or lookups required), but it has security tradeoffs: because the real digit's position is constant, attackers who gain partial knowledge about memory layout, instrumentation hooks, or debug output may eventually learn the convention and target the last slot.

To harden this approach, store only hashed representations of the full ordered bucket (e.g., HMAC over the ordered list plus salt) rather than raw digits, perform the extraction and comparison in constant time to reduce timing side-channels, and validate all inputs to prevent injection or bucket-tampering. Additionally, combine the fixed-index rule with complementary protections such as per-user salts, rate limiting, tamper-resistant key storage (HSM) for any selector material, and logging/alerting on anomalous verification attempts; this preserves the simplicity of the "last element" extraction while mitigating many of the predictable weaknesses that arise from a fully static selection rule.

8. APPLICATIONS

The Multi-Digit Bucket Hashing technique can be applied across a wide range of security-critical systems that rely on PIN authentication. In ATM infrastructures, it provides a more resilient method for storing and handling customer PINs, reducing the risk posed by memory scraping or reverse-engineering of ATM software. Banking and mobile payment applications can adopt this approach to protect PIN-based login or transaction verification processes, ensuring that even if an attacker gains access to device memory, the actual digits remain obscured. Secure PIN pads and digital lock systems can use bucket-based representations to safeguard local PIN processing, making physical or side-channel attacks far less effective. Likewise, smart card authentication benefits from this model by preventing leakage of PIN digits through predictable storage or processing patterns. Finally, because the method is computationally lightweight, it is well suited for resource-constrained environments and can be integrated into lightweight cryptographic systems where security must be achieved without heavy computational overhead.

8.1 Threat Model

A comprehensive threat model is essential to understand the adversarial landscape. The proposed system assumes the presence of passive and active attackers such as:

- **Memory Forensics Attacker:** extracts RAM snapshots to locate PIN structures.
- **Reverse Engineering Attacker:** analyzes the application code and memory flow.
- **Side-Channel Attacker:** attempts to detect digit-wise processing patterns.
- **Database Intrusion Attacker:** retrieves stored PIN-related structures.

The model defends against these attackers by introducing confusion, misdirection, and multi-digit noise that complicate identification of real PIN digits.

8.2 Security Analysis

The embedding of random digits into each bucket significantly strengthens the security posture:

- **Confusion:** Real digits are indistinguishable among noise digits.
- **Unpredictability:** bucket sizes and noise digits vary per user/session.
- **Resistance to Memory Extraction:** memory dumps display multiple digits, not a clear PIN.
- **Key-Dependent Retrieval:** Only systems with the retrieval rule can reconstruct the PIN.
- **Statistical Resistance:** attackers cannot perform frequency analysis because noise digits change each run.

This structure increases computational cost for attackers while maintaining low cost for legitimate users.

8.3 Complexity Analysis

Time Complexity

- **Packing:** $O(n \times k)$ where n = PIN length, k = noise digits.
- **Unpacking:** $O(n)$, since retrieving the real digit is constant-time per bucket.

Space Complexity

- **Total Storage:** $O(n \times k)$ compared to $O(n)$ in traditional PIN models.

The slight increase in memory usage is negligible for modern systems.

8.4 Literature Review

Several works explore PIN protection and obfuscation, but none adopt the bucket-hashing approach:

- Traditional methods rely on **hashing**, which still preserves 4-digit structure.
- Homomorphic encryption protects operations but is computationally expensive.
- Homophonic substitution encodes symbols with multiple outputs, similar in concept but not data-structure oriented.
- Data-structure obfuscation literature explores fake pointers and memory padding but not specifically PIN digits.

The proposed method fills a gap by combining hash bucket design, cryptographic confusion, and PIN obfuscation.

8.5 Comparison with Existing Approaches

Method	Advantages	Limitations
Standard Hashing	Fast, simple	Predictable structure, easy memory extraction
AES Encryption	Strong security	Heavy for embedded systems
Homomorphic Encryption	Operable on encrypted data	Very heavy computational cost
Proposed Bucket Method	Lightweight, adds confusion, resists extraction	Requires secure retrieval rule

8.6 Integration with Rook Tour Method

The bucket retrieval rule can be enhanced using a Rook Tour traversal:

- The chessboard traversal sequence determines the extraction index.

- Increases randomness and key-dependency.
- Makes bucket decoding impossible without Rook Tour key.

This hybrid model offers stronger obfuscation suitable for cryptographic frameworks.

9. CONCLUSION

The proposed multi-digit hash bucket technique offers a lightweight yet powerful means of obscuring PIN digits. By embedding each real digit alongside random noise values, attackers cannot identify the correct digit without prior knowledge of the selection rule. The model enhances confusion, protects against memory forensics, and is compatible with existing PIN systems.

10. FUTURE WORK

- Integrating Rook Tour traversal to determine real-digit selection index.
- Adding encryption to noise digits.
- Expanding bucket size dynamically.
- Implementing hardware acceleration for ATM cards.
- Creating a hybrid scheme combining AES + bucket obfuscation.

REFERENCES

1. Shannon, C. E. (1949). Communication theory of secrecy systems. Bell System Technical Journal, 28(4), 656–715.
2. Diffie, W., & Hellman, M. (1976). New directions in cryptography. IEEE Transactions on Information Theory, 22(6), 644–654.
3. Rivest, R. L., Shamir, A., &Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120–126.
4. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
5. Knuth, D. E. (1998). The art of computer programming: Volume 3 – Sorting and searching (2nd ed.). Addison–Wesley.
6. Carter, J. L., &Wegman, M. N. (1979). Universal classes of hash functions. Journal of Computer and System Sciences, 18(2), 143–154.
7. Aviv, A. J., Sapp, B., Wolf, M., & Blaze, M. (2010). Smudge attacks on smartphone touch screens. Proceedings of the 4th USENIX Workshop on Offensive Technologies.
8. Bonneau, J., Herley, C., Van Oorschot, P. C., &Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of Web authentication schemes. IEEE Symposium on Security and Privacy.
9. Federal Financial Institutions Examination Council (FFIEC). (2011). Authentication in an Internet banking environment. FFIEC Guidance Document.
10. Florêncio, D., &Herley, C. (2007). A large-scale study of web password habits. Proceedings of the 16th International Conference on World Wide Web.
11. Bonneau, J. (2012). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. IEEE Symposium on Security and Privacy.
12. Collberg, C., &Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation—Tools for software protection. IEEE Transactions on Software Engineering, 28(8), 735–746.
13. Sion, R. (2007). Secure data outsourcing and obfuscation. ACM International Conference on Computer Systems.
14. Furnell, S. M., & Clarke, N. L. (2012). Human factors and authentication: Challenges for multi-factor mechanisms. International Journal of Human-Computer Studies, 70(10), 823–834.
15. De Luca, A., Hang, A., Brudy, F., Lindner, C., & Hussmann, H. (2012). Touch me once and I know it's you!: Implicit authentication based on touch screen patterns. ACM SIGCHI Conference on Human Factors in Computing Systems.
16. National Institute of Standards and Technology (NIST). (2012). Secure Hash Standard (SHS) (FIPS PUB 180-4).
17. National Institute of Standards and Technology (NIST). (2017). Digital Identity Guidelines (SP 800-63B).